Session 121

Carbon Performance Tuning

John Iarocci
Manager, Advanced Mac Toolbox

Why Is My App Slow?

- Basic problem
  - Some APIs are not performance compatible with Mac OS 9
  - Code assumes compatibility
- Many answers...

Performance Menu

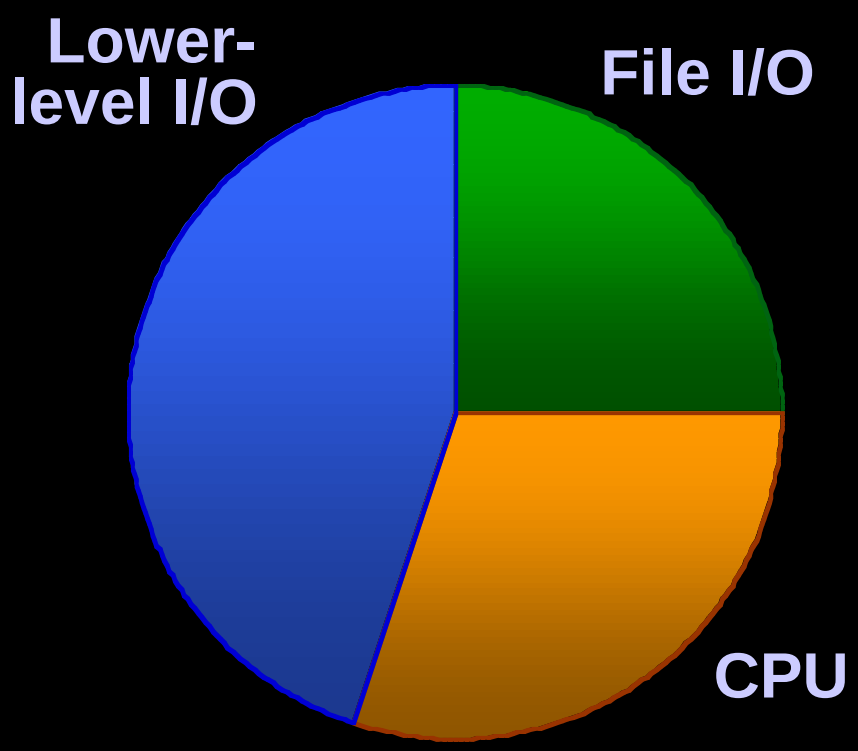
- Application launch
- File System usage
- CPU usage
- Application responsiveness
- Memory usage
- Resource use
- Code loading
- Polling vs. blocking
- Threading
- Asynchronous I/O

Application Launch

Application Launch

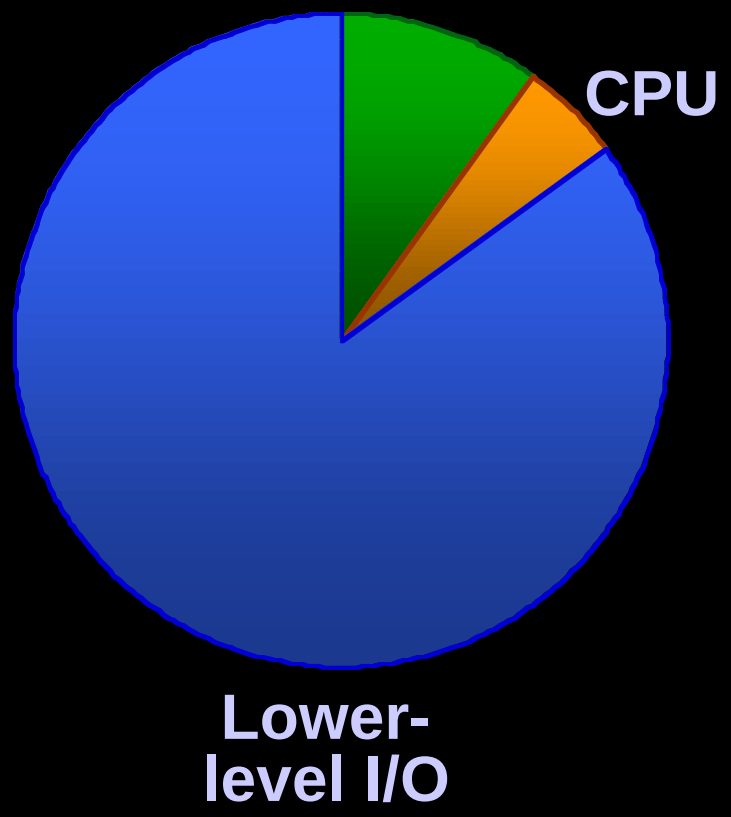
- Two different kinds
  - Cold launch
  - Warm launch
  - Challenge: 1 bounce!
- Measurement Techniques
  - The “do nothing” app
  - The “best case situation”

Launch Profile—Untuned



- File I/O
- Pauses
- Writing
- Excessive computation

Launch Profile—Tuned



- Low-level I/O
- Minimal computation
- No writes
- Minimal file I/O
- No pauses

Launch Speed-ups

- Concentrate on:
  - CPU usage
  - File system usage
- Do only what you need to
- Eliminate
  - Writes and pauses
  - Dead-code and redundant I/O

Session 121

File System Performance

Nitin Ganatra
File Systems Tech Lead

File Performance Topics

- File iteration/metadata
- Volume iteration
- File I/O
- Writing large files
- File assumptions

File Iteration/Metadata

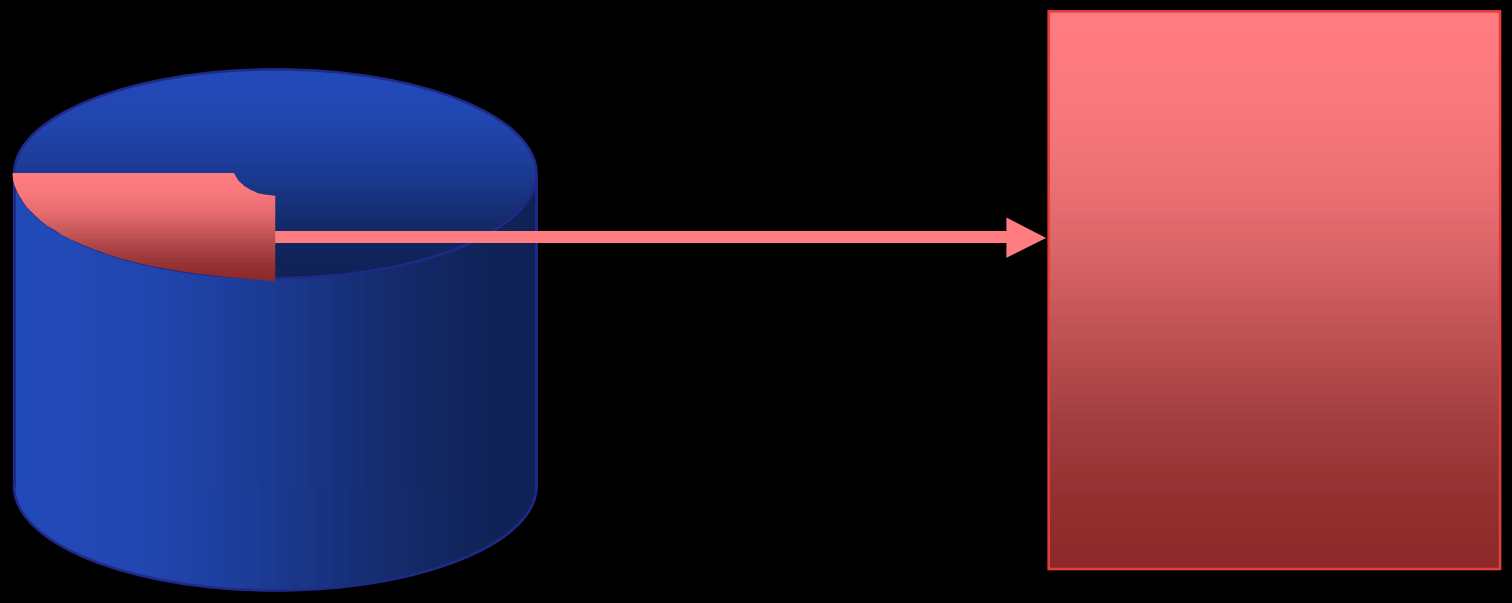
- PBGetCatInfo
  - Not optimal on any file system
  - Overkill for most clients

GetCatInfo ParamBlock

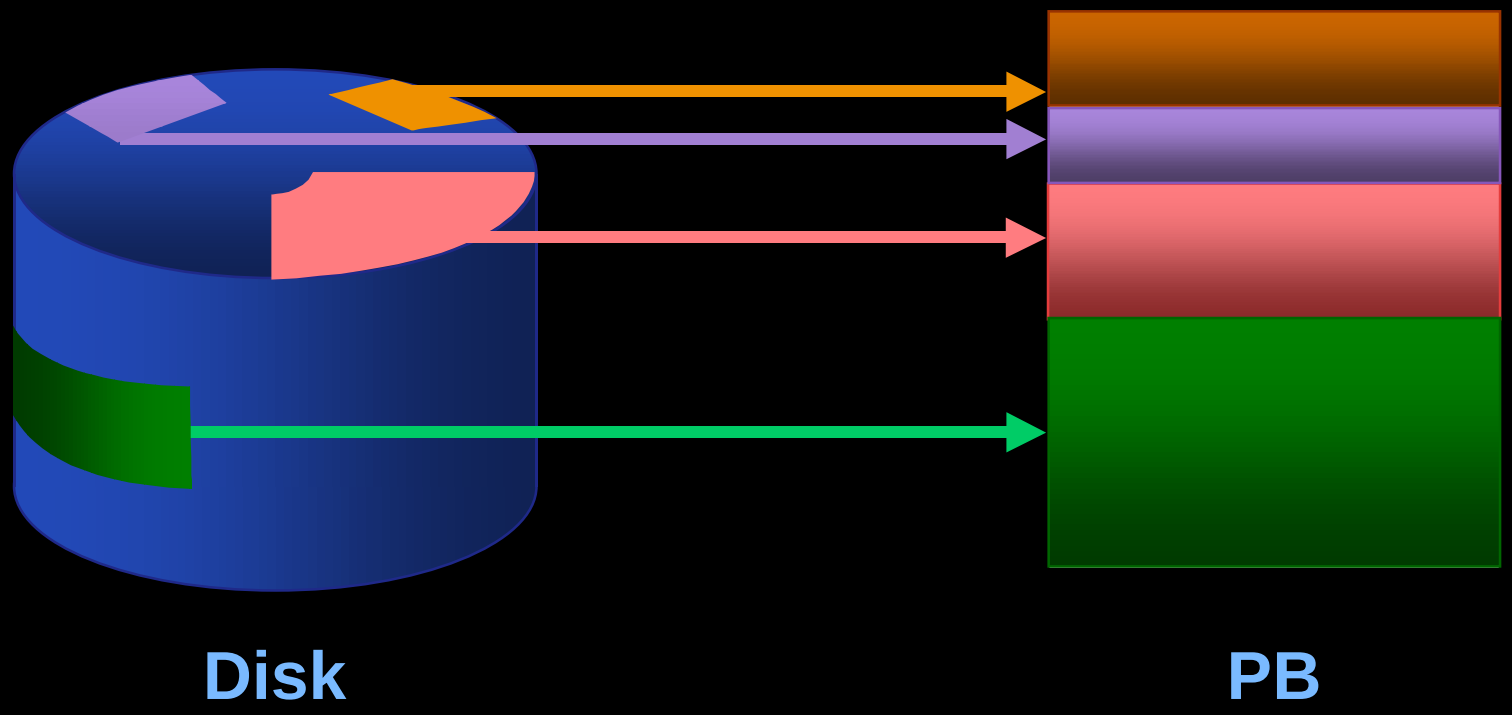
- VRefNum
- DirID
- Parent DirID
- Name
- Flags
- FileRefNum
- Clump size
- Access rights
- Directory Valence
- Logical data/resource fork lengths
- Physical data/resource fork lengths
- FinderInfo, extended FinderInfo
- Creation date, Modification date, backup date…

and more…

PBGetCatInfo—HFS/AFP



PBGetCatInfo—WebDAV/nfs/Samba



FSGetCatalogInfo

- Allows the OS to optimize
- Only ask for what you need!
  - Use fs_usage

FSGetCatalogInfo Sample

```
Boolean IsFolder (const FSRef *ref)
{
  err = FSGetCatalogInfo(ref,
  kFSCatInfoNodeFlags, &info, NULL, …);

  return (info.nodeFlags & ioDirMask);
}
```

Volume Iteration

- Not free in Carbon
  - No more direct VCB access
- PBHGetVInfo is expensive
- Use FSGetVolumeInfo

File I/O

- File manager performance and caching
  - **http://developer.apple.com/technotes/fl/fl_16.html**
- Large, page-aligned I/O
  - Use fs_usage
- Don't pollute the cache
  - noCacheMask

Writing Large Files

- SetEOF/FSSetForkSize
- Use PBAllocate, or just write

File Assumptions

- User Preferences
  - May be network-based
- Set up ‘hostile’ test environment
  - NFS user home directory
  - Mac OS X Server documentation

Session 121

More Performance

John Iarocci
Manager, Advanced Mac Toolbox

CPU Usage

- Preemptive != “free CPU”
- Limited resource
- Use threads and timers wisely
- Tools: *top, time, CPU Monitor*

Responsiveness I

- Event handling
  - Use Tracking APIs instead of polling
    - i.e., **TrackMouseLocation()** instead of **StillDown() or Button()**
    - Tools: Sampler, CPU Monitor
- Drawing
  - Eliminate redundant operations
  - Use **QDIsPortBufferred()**
  - Tools: QuartzDebug, Sampler

Responsiveness II

- Flushing
  - Try to avoid
  - Okay for animations and splash screens
- Window Resizing
  - Use SetWindowBounds()
  - Invisible Windows
- File I/O during UI operations
  - Tools: fs_usage, Sampler

Polling vs. Blocking

- WaitNextEvent(0)
- Time—TickCount()
- Volumes
- Preference change notifications
- We need notification too:
  - Please call FNNotify()

Resource Mgr Use

- Avoid small reads and writes
- Avoid opening/closing resource files
- Avoid unnecessary **UpdateResFile()** calls
- Set the **CSResourcesFileMapped** key in your application's plist
  - See Tech Note #2013

Memory Usage

- Leaks
- Size of application (code, data)
- Handles vs. Pointers
- HLock() & HUnlock()
  - Still needed on Mac OS 9
- Purging memory and fixed heaps
- Tools: *leaks, MallocDebug*

Code Loading

- Mac OS X let's you "pay as you go"
- Factor your app
  - Dynamic loading
  - Use plug-ins for optional features
- Reduce number of libraries used

Asynchronous I/O

- Completion Routines
  - Added cost on Mac OS X
  - Constrained runtime
- Time Mgr
  - Not efficient for custom threading packages

Threading Problems

- Cooperative Threads
  - Not CPU friendly
  - Worse when combined with async I/O
- Messaging between threads
- Too many threads

Threading Solutions

- MP Threads
  - Based on pthreads
- Useful for:
  - “Safe Save” feature
  - Network listener
  - Low-priority computation

Summary

- Performance is a feature!
- Lather, rinse, repeat…
- Become an expert with the Mac OS X perf tools
- Go after those performance problems

More Information

- Carbon Developer Documentation
  http://developer.apple.com/techpubs/carbon/carbon.html

- Mac OS X Performance
  http://developer.apple.com/techpubs/macosx/SystemOverview/Performance/Performance.pdf

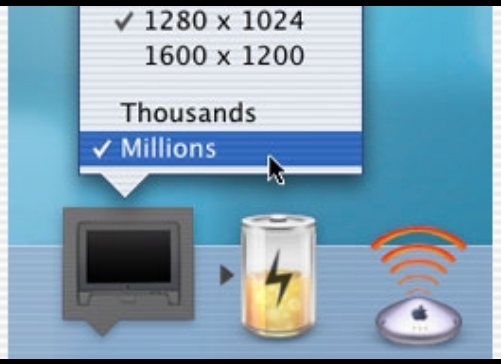
Roadmap

705 Apple Performance Tools	Room A2 **Thurs., 5:00pm**
106 Carbon Event Manager	Hall 2 **Tues., 5:00pm**
112 Carbon Windows & Menus	Hall 2 **Wed., 2:00pm**
120 Carbon Controls & Appearance II	Hall 2 **Thurs., 10:30am**

Who to Contact

Mark Turner
Carbon Technology Manager
Apple Worldwide Developer Relations
mark@apple.com

Q&A



Mark Turner
Carbon Technology Manager
mark@apple.com

http://developer.apple.com/events/urls.html

WWDC
Worldwide Developers Conference 2001